Tipologia y Ciclo de Vida de los Datos

PRAC2 - Integración, Limpieza, Validación y Análisis de los Datos

Autor: Leidy Liliana Torres Bolívar & Jose Carlos Sola Verdú

Diciembre 2020

1. Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

Para la realización de este ejercicio se usará un conjunto de datos que contiene los clientes de un distribuidor mayorista. Incluye el gasto anual en unidades monetarias (mu) en diversas categorías de productos (Lacteos, carnes, congelados ETC) Se evidencia que son datos que se pueden trabajar en problemas no supervisados con métodos de agregación (clusters) ya que no cuentan con una categorización o clase.

Fuente. https://www.kaggle.com/binovi/wholesale-customers-data-set

2. Integración y selección de los datos de interés a analizar

```
# Como primer paso se llaman las librerias necesarias para el ejercicio

library(ggplot2)
library(dplyr)
library(dummies)
library(kableExtra)
library(scales)
library(cluster)
library(factoextra)
library(NbClust)
library(arules)
library(fpc)

#Se carga el conjunto de datos el cual contiene los nombres de sus atribu
tos.

customers_data<-read.csv("https://archive.ics.uci.edu/ml/machine-learning
-databases/00292/Wholesale%20customers%20data.csv", header=T, sep=",")
```

```r
attach(customers_data)

#Visualizamos el data frame
head(customers_data)
```

```
##   Channel Region Fresh Milk Grocery Frozen Detergents_Paper Delicassen
## 1       2      3 12669 9656    7561    214             2674       1338
## 2       2      3  7057 9810    9568   1762             3293       1776
## 3       2      3  6353 8808    7684   2405             3516       7844
## 4       1      3 13265 1196    4221   6404              507       1788
## 5       2      3 22615 5410    7198   3915             1777       5185
## 6       2      3  9413 8259    5126    666             1795       1451
```

Descripción de las diferentes variables:

FRESH: annual spending (m.u.) on fresh products (Continuous)

MILK: annual spending (m.u.) on milk products (Continuous)

GROCERY: annual spending (m.u.)on grocery products (Continuous)

FROZEN: annual spending (m.u.)on frozen products (Continuous)

DETERGENTS_PAPER: annual spending (m.u.) on detergents and paper products (Continuous)

DELICATESSEN: annual spending (m.u.)on and delicatessen products (Continuous)

CHANNEL: 1=Horeca 2=Retail

REGION: 1=Lisnon, 2=Oporto or 3=Other

```r
#Se valida la estructura de los datos
 str(customers_data)
```

```
## 'data.frame':    440 obs. of  8 variables:
##  $ Channel         : int  2 2 2 1 2 2 2 2 1 2 ...
##  $ Region          : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ Fresh           : int  12669 7057 6353 13265 22615 9413 12126 7579
5963 6006 ...
##  $ Milk            : int  9656 9810 8808 1196 5410 8259 3199 4956 3648
11093 ...
##  $ Grocery         : int  7561 9568 7684 4221 7198 5126 6975 9426 6192
18881 ...
##  $ Frozen          : int  214 1762 2405 6404 3915 666 480 1669 425 115
9 ...
##  $ Detergents_Paper: int  2674 3293 3516 507 1777 1795 3140 3321 1716
7425 ...
##  $ Delicassen      : int  1338 1776 7844 1788 5185 1451 545 2566 750 2
098 ...
```

3. Limpieza de los datos

3.1 ¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?

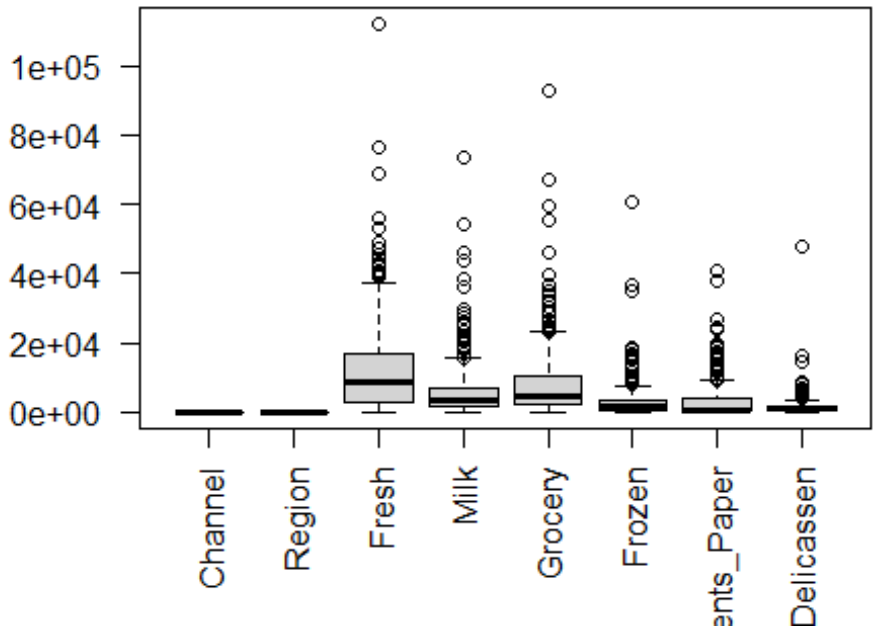
```
#Se valida que no existan valores vacios
colSums(is.na(customers_data))
```

```
##          Channel           Region            Fresh             Milk 
##                0                0                0                0 
##          Grocery           Frozen Detergents_Paper       Delicassen 
##                0                0                0                0
```

Se evidencia que el set de datos no contempla valores vacíos, si embargo en caso de tener valores vacios en valores contunios se usaria la media de conjunto de datos para completa dicha información.

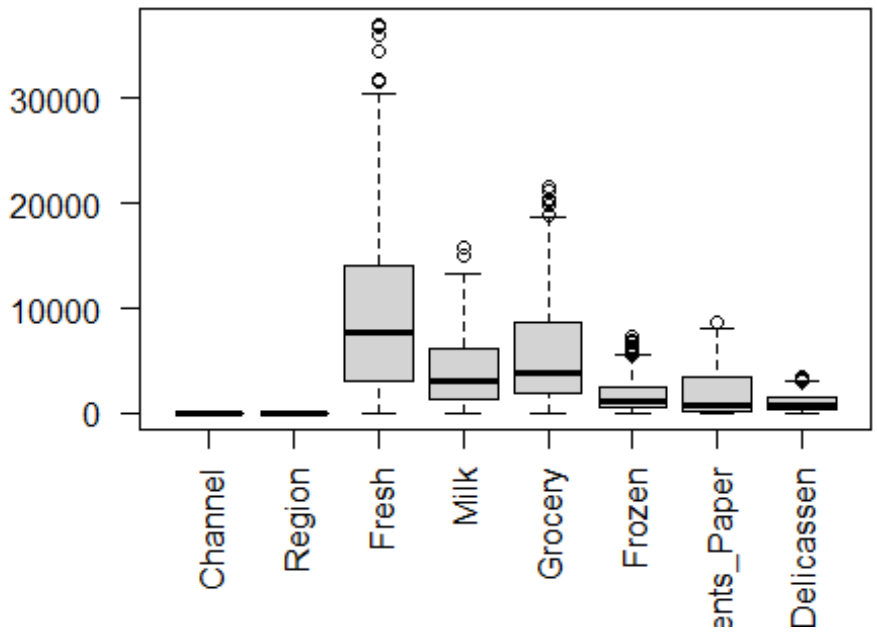
3.2 Identificación y tratamiento de valores extremos

```
#Se hace validación de Outliers por medio de la gráfica boxplot para cada una de las variables
boxplot(customers_data,las=2)
```



Como se observa hay Outliers en las variables Fresh, Milk, Grocery, Frozen, Detergents_Paper y Delicassen, lo cual podría claramente la clasificación de los datos, por eso se descartarán usando la diferencia de los percentiles 25 y 75 y calculando el IQR (Rango intercuatilico) para encontrar los rangos y descartar valore atipicos.

```
#Quitando datos atípicos de las variables Fresh,Milk,Grocery,Frozen,Deter
gents_Paper,Delicassen almacenandola en el df customers_data2
customers_data2 <- customers_data
for (x in colnames(customers_data2[3:8])){
  qt <- quantile(customers_data[[x]], probs=c(.25, .75), na.rm = FALSE)
  iqr <- IQR(customers_data2[[x]])
  superior <-  qt[2]+1.5*iqr
  inferior<- qt[1]-1.5*iqr
  customers_data2<- subset(customers_data2, customers_data2[[x]] > (qt[1]
- 1.5*iqr) & customers_data2[[x]] < (qt[2]+1.5*iqr))
}
#Se grafica nuevamente los valores atipicos usando el nuevo df
boxplot(customers_data2, las=2)
```



Comparando las dos gráficas se observa que el número de valores atipicos bajó considerablente respecto al DF original, esto sin duda ayudará a mejorar el rendieminto del modelo de agregación.

```r
#Normalización de datos para manejar la misma escala en las diferentes va
riables
customers_data_scale <- scale(customers_data2)
head(customers_data_scale)
```

```
##      Channel   Region       Fresh       Milk    Grocery      Frozen
## 1  1.6065098 0.576818  0.36757080  1.6490562  0.3811910 -0.97640156
## 2  1.6065098 0.576818 -0.31831375  1.6947119  0.7958942 -0.02850308
## 4 -0.6205403 0.576818  0.44041242 -0.8590425 -0.3089478  2.81396769
## 6  1.6065098 0.576818 -0.03036934  1.2348938 -0.1219491 -0.69962500
## 7  1.6065098 0.576818  0.30120670 -0.2652220  0.2601068 -0.81351978
## 8  1.6065098 0.576818 -0.25451622  0.2556680  0.7665530 -0.08545047
##   Detergents_Paper Delicassen
## 1       0.33835403  0.4390311
## 2       0.60961495  0.9896503
## 4      -0.61127831  1.0047357
## 6      -0.04684524  0.5810858
## 7       0.54256662 -0.5578661
## 8       0.62188524  1.9827761
```

4. Análisis de los datos

4.1 Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar)

En primer lugar, se realizará un resumen estadístico sobre las variables que se van a utilizar:

```r
summary(customers_data)
```

```
##     Channel          Region          Fresh             Milk
##  Min.   :1.000   Min.   :1.000   Min.   :     3   Min.   :   55
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.:  3128   1st Qu.: 1533
##  Median :1.000   Median :3.000   Median :  8504   Median : 3627
##  Mean   :1.323   Mean   :2.543   Mean   : 12000   Mean   : 5796
##  3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.: 16934   3rd Qu.: 7190
##  Max.   :2.000   Max.   :3.000   Max.   :112151   Max.   :73498
##     Grocery          Frozen        Detergents_Paper    Delicassen
##  Min.   :    3   Min.   :   25.0   Min.   :    3.0   Min.   :    3.0
##  1st Qu.: 2153   1st Qu.:  742.2   1st Qu.:  256.8   1st Qu.:  408.2
##  Median : 4756   Median : 1526.0   Median :  816.5   Median :  965.5
##  Mean   : 7951   Mean   : 3071.9   Mean   : 2881.5   Mean   : 1524.9
##  3rd Qu.:10656   3rd Qu.: 3554.2   3rd Qu.: 3922.0   3rd Qu.: 1820.2
##  Max.   :92780   Max.   :60869.0   Max.   :40827.0   Max.   :47943.0
```

Seguidamente, se seleccionaran los grupos del conjunto de datos que pueden resultar interesantes para analizar/comparar:

```
# Agrupación por la Región dónde se han vendido los productos (variable R
egion)

customers_data.Vendido_en_Lisboa <- customers_data[customers_data$Region
== "1",]
customers_data.Vendido_en_Oporto <- customers_data[customers_data$Region
== "2",]
customers_data.Vendido_en_Otros <- customers_data[customers_data$Region =
= "3",]


# Agrupación por Ventas al por Mayor o al por Menor (variable Channel)

customers_data.porMayor <- customers_data[customers_data$Channel == "1",]
customers_data.porMenor <- customers_data[customers_data$Channel == "2",]
```

Una vez terminada la fase de análisis y preparación de los datos. En esta fase se usaran diferentes métodos para identificar el número de clusters más óptimo, Se usará el metodo de la silueta media.

4.2 Comprobación de la normalidad y homogeneidad de la varianza

Para la comprobación de que los valores de las variables cuantitativas están distribuidas normalmente, se hará uso de la prueba de normalidad de *Anderson-Darling.*

De esta forma se comprueba que cada variable obtiene un p_valor superior a alpha = 0.05. Si se cumple esta condición, se considerará que una determinada variable sigue una distribución normal.

```
library (nortest)

alpha = 0.05

col.names = colnames (customers_data)

for (j in 1:ncol (customers_data)) {
  if (j == 1) cat ("Variables que no siguen una distribución normal:\n")
  if (is.integer (customers_data [ , j]) | is.numeric (customers_data [ ,
j])) {
    p_val = ad.test (customers_data [ , j]) $p.value
    if (p_val < alpha) {
      cat (col.names[j])

      # Format output
      if (j < ncol (customers_data) - 1) cat (", ")
      if (j %% 3 == 0) cat ("\n")
```

```
    }
  }
}

## Variables que no siguen una distribución normal:
## Channel, Region, Fresh,
## Milk, Grocery, Frozen,
## Detergents_PaperDelicassen
```

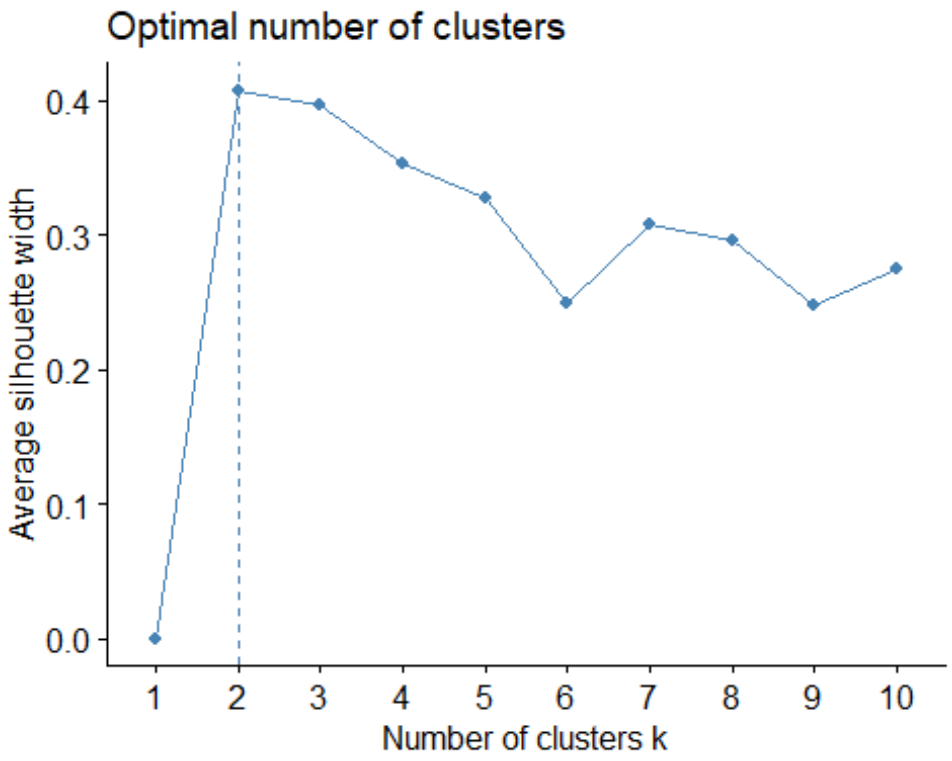
El siguiente paso será comprobar la homogeinidad de varianzas. Para ello, se utilizará la aplicación de test de *Fligner-Killeen*. Se estudiará la homogeinidad entre diferentes grupos formados por la region y el tipo de venta (channel). También se puede estudiar la homogeinidad entre las distintas variables del tipo de alimentos.

```
fligner.test(Channel ~ Region, data = customers_data)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Channel by Region
## Fligner-Killeen:med chi-squared = 4.3393, df = 2, p-value = 0.1142

fligner.test(Fresh ~ Milk, data = customers_data)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Fresh by Milk
## Fligner-Killeen:med chi-squared = 439, df = 420, p-value = 0.2518

fligner.test(Grocery ~ Frozen, data = customers_data)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Grocery by Frozen
## Fligner-Killeen:med chi-squared = 439, df = 425, p-value = 0.3093

fligner.test(Detergents_Paper ~ Delicassen, data = customers_data)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Detergents_Paper by Delicassen
## Fligner-Killeen:med chi-squared = 408.81, df = 402, p-value = 0.3965
```

Como se puede apreciar, todas las comparaciones superan el p_valor de 0.05, por lo que se entiende que la hipótesis de las varianzas son homogéneas.

5. Aplicación de pruebas estadísticas para comparar los grupos de datos y Representación de los resultados a partir de tablas y gráficas

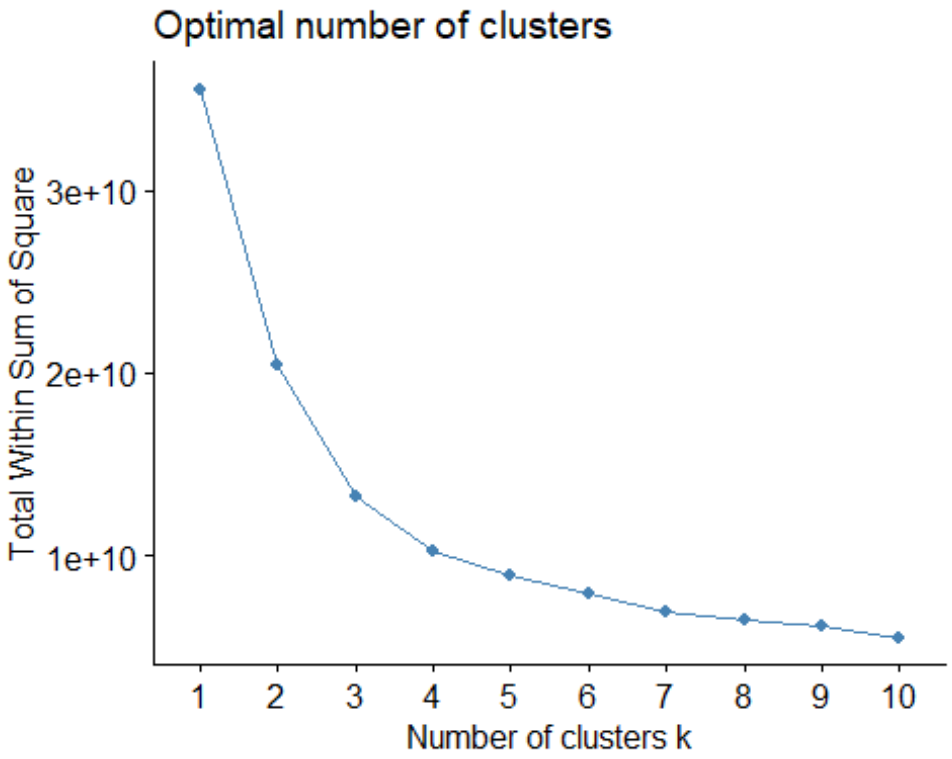
```
#El método de la silueta media el cual mide la distancia de separación en
tre los clústers. Nos indica como de cerca está cada punto de un clúster
a puntos de los clústers vecinos.

fviz_nbclust(customers_data2, kmeans, method="silhouette")
```



Según la grafica presentada anteriormente el mejor número de clusters serían k=2

Ahora usando el método wss de K-Means.

```
#Ahora usando el método wss.
fviz_nbclust(customers_data2, kmeans, method="wss")
```



Al usar el metodo anterior se observa que la recomendación de clusters serían entre 3 y 4.

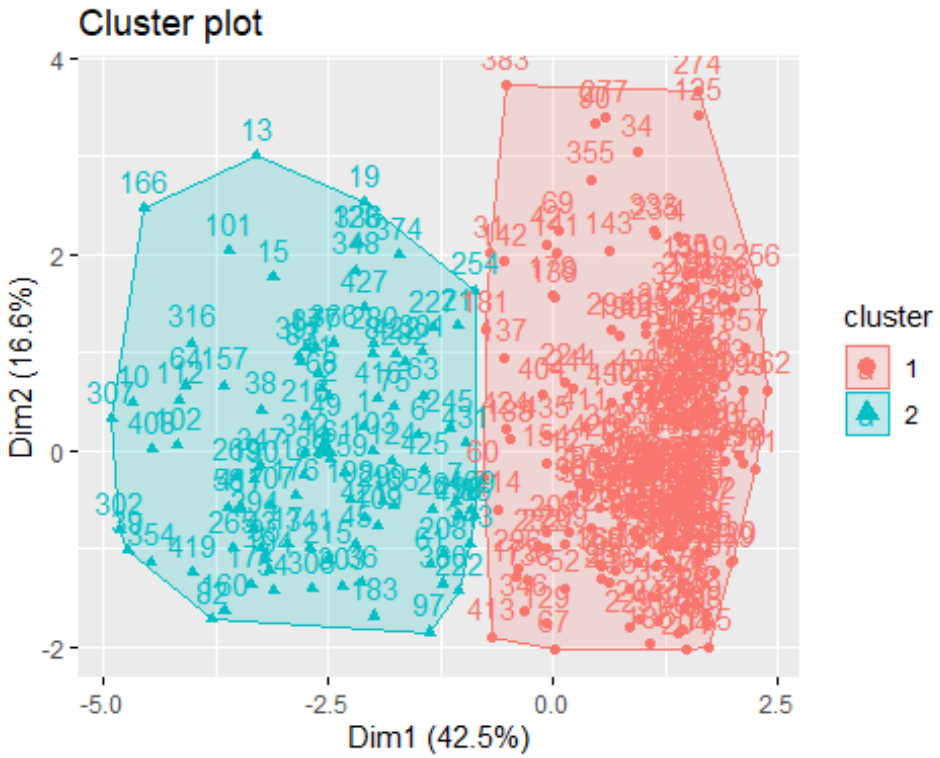
Ahora usamos la función *kmeansruns* del paquete *fpc* que ejecuta el algoritmo *kmeans* con un conjunto de valores, para después seleccionar el valor del número de clústers que mejor funcione de acuerdo a dos criterios: la silueta media ("asw") y *Calinski-Harabasz* ("ch"). Usado en el ejercicio de ejemplo de la base de datos *irs*.

```
fit_ch  <- kmeansruns(customers_data2, krange = 1:10, criterion = "ch")
fit_asw <- kmeansruns(customers_data2, krange = 1:10, criterion = "asw")

#asw la silueta media
fit_ch$bestk

## [1] 3

#ch Calinski-Harabasz
fit_asw$bestk

## [1] 2
```

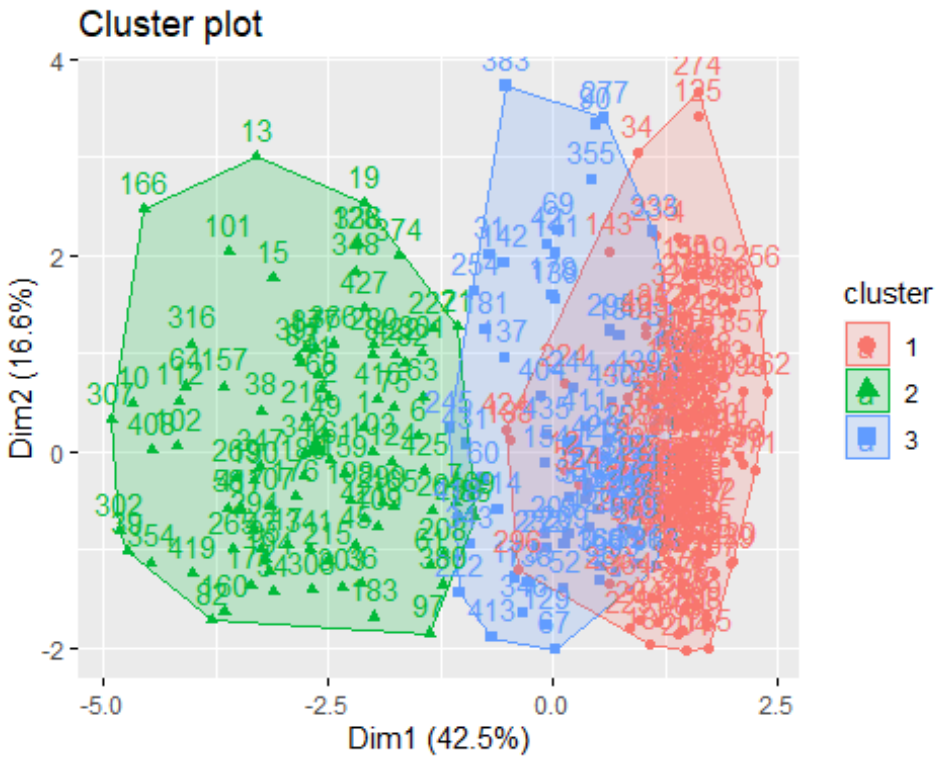
Según los resultados presentados anteriormente asw y ch recomiendan entre 2 y 3 clusters respectivamente.

Como los resultados del número de clusters estan entre dos y tres probaremos el algoritmo de K-Means con 2 y 3 para ver cual sería el más óptimo.

```r
#Con 2 clusters
k2_scale <- kmeans(customers_data_scale, 2)

#Graficamos los clusters generados con fviz_cluster
fviz_cluster(k2_scale, data = customers_data_scale)
```




```r
#Con 3 clusters
k3_scale <- kmeans(customers_data_scale, 3)

#Graficamos los clusters generados con fviz_cluster
fviz_cluster(k3_scale, data = customers_data_scale)
```



Como se observa en las gráficas generadas con *fviz_cluster* claramente el mejor número de clusters es 2 ya que se puede observar claramente la distinción entre los dos grupos, mientras que en la gráfica con 3 clusters hay algunos puntos que se solapan.

Ahora vamos a identificar las caracteristicas de cada uno de los clusters identificando de a dos variables la distribución de los dos clusters

```
summary(customers_data2)
```

```
##     Channel          Region          Fresh             Milk
##  Min.   :1.000   Min.   :1.000   Min.   :    3   Min.   :   55
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.: 3026   1st Qu.: 1365
##  Median :1.000   Median :3.000   Median : 7769   Median : 3086
##  Mean   :1.279   Mean   :2.557   Mean   : 9661   Mean   : 4094
##  3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.:14070   3rd Qu.: 6248
##  Max.   :2.000   Max.   :3.000   Max.   :37036   Max.   :15729
##     Grocery          Frozen        Detergents_Paper   Delicassen
##  Min.   :    3   Min.   :  25.0   Min.   :   3.0   Min.   :   3.0
##  1st Qu.: 2003   1st Qu.: 575.5   1st Qu.: 229.5   1st Qu.: 353.5
##  Median : 3823   Median :1206.0   Median : 706.0   Median : 749.0
##  Mean   : 5716   Mean   :1808.5   Mean   :1901.9   Mean   : 988.8
##  3rd Qu.: 8699   3rd Qu.:2558.0   3rd Qu.:3379.5   3rd Qu.:1445.5
##  Max.   :21570   Max.   :7332.0   Max.   :8752.0   Max.   :3508.0
```

```
k2 <- kmeans(customers_data2, 2)
```

```
k2
```

```
## K-means clustering with 2 clusters of sizes 228, 95
##
## Cluster means:
##    Channel   Region     Fresh     Milk  Grocery   Frozen Detergents_Pa
per
## 1 1.307018 2.561404  5283.417 4335.754 6031.583 1682.211         2198.
873
## 2 1.210526 2.547368 20168.853 3512.474 4959.232 2111.758         1189.
158
##   Delicassen
## 1   893.0965
## 2  1218.3684
##
## Clustering vector:
##   1   2   4   6   7   8   9  10  11  12  13  14  15  16  17  19  20  2
1  22  26
##   1   1   2   1   1   1   1   1   1   2   2   2   2   1   1   2   1
2   1   2
##  27  28  31  32  33  34  35  36  38  39  42  43  45  49  51  52  54  5
5  56  58
##   1   2   2   1   2   2   1   1   2   1   2   1   1   1   1   1   1
2   1   1
##  59  60  61  63  64  65  67  68  69  70  75  76  79  80  81  82  83  8
4  85  90
##   2   1   1   1   1   1   1   2   1   1   1   2   1   1   1   1   1
2   1   2
##  91  95  96  97  98  99 100 101 102 103 105 106 107 109 111 112 114 11
5 116 117
##   1   1   1   1   1   1   1   1   1   1   2   2   1   1   1   1   2
2   1   1
## 118 119 120 121 122 123 124 125 128 129 131 132 133 134 135 136 137 13
8 139 140
##   1   2   1   2   1   2   1   2   2   1   1   1   2   1   1   1   1
1   2   1
## 141 142 143 145 147 148 149 150 151 152 153 154 155 157 158 159 160 16
1 162 163
##   2   2   2   2   1   1   1   2   2   1   2   1   1   1   2   1   1
1   2   2
## 165 166 168 169 170 171 173 175 176 178 179 180 181 183 185 186 187 18
8 189 190
##   1   2   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1
1   1   1
## 191 192 193 194 195 198 199 200 204 205 207 208 209 211 213 214 215 21
6 218 220
##   2   2   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1
1   2   1
## 221 222 224 225 226 227 228 229 230 232 233 234 235 236 237 238 239 24
2 243 244
##   2   1   1   1   2   2   1   1   1   1   2   1   2   1   1   2   1
2   2   1
```

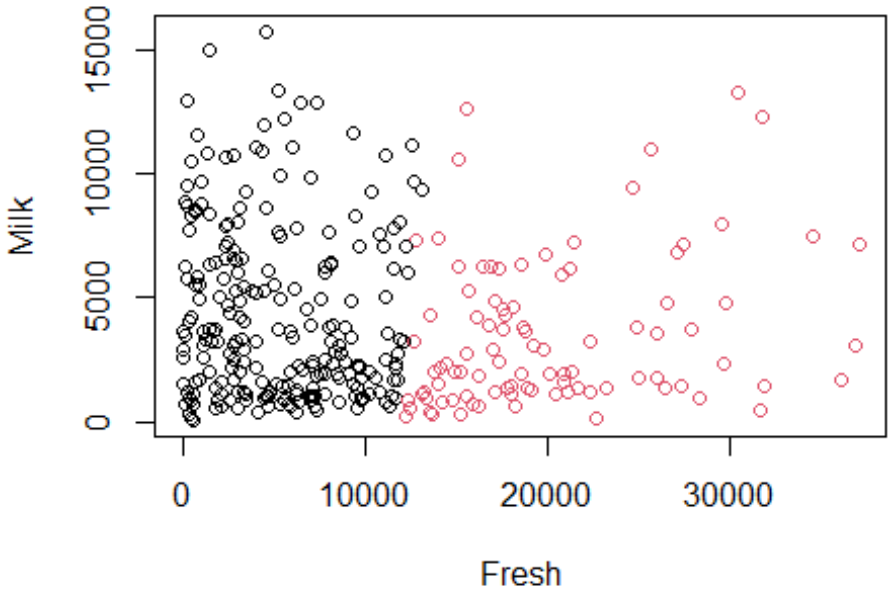
```
## 245 247 248 249 250 251 254 256 257 258 261 262 263 264 265 269 270 27
1 272 273
##   1   1   2   2   1   1   2   2   1   1   1   1   2   1   1   1   2
1   1   1
## 274 275 276 277 279 280 281 282 287 289 291 292 293 294 295 296 297 29
8 299 300
##   2   1   1   2   1   2   1   1   1   2   1   1   1   1   2   1   2
1   1   1
## 301 302 303 306 307 308 309 312 314 315 316 317 318 319 321 322 323 32
4 325 327
##   2   1   1   1   1   2   1   2   1   1   1   1   1   1   1   1   2
2   2   1
## 328 330 331 333 336 337 341 342 343 345 346 347 348 349 351 353 354 35
5 356 357
##   1   1   1   2   2   2   1   1   1   1   1   1   2   1   1   1   1
2   1   2
## 360 361 362 363 364 365 366 367 368 369 370 374 375 376 377 379 380 38
1 383 384
##   1   2   1   1   1   1   1   1   1   2   1   2   1   1   1   1   1
2   2   1
## 386 387 388 389 390 391 392 393 395 396 397 398 399 400 401 403 404 40
5 406 408
##   1   1   2   1   1   1   1   1   2   1   1   1   1   1   1   2   2
2   1   1
## 409 411 413 415 416 417 418 419 420 421 422 423 424 425 427 429 430 43
1 433 434
##   1   1   1   1   1   1   1   1   1   1   2   2   2   2   1   1   1
1   2   1
## 435 439 440
##   2   1   1
##
## Within cluster sum of squares by cluster:
## [1] 13858574305  6574079992
##  (between_SS / total_SS =  42.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.w
ithinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
str(k2)
```

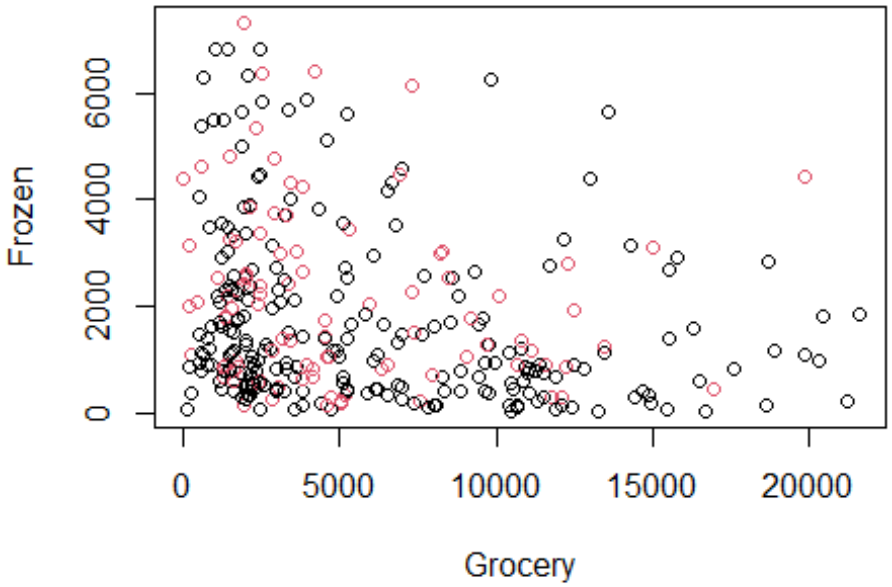
```
## List of 9
##  $ cluster     : Named int [1:323] 1 1 2 1 1 1 1 1 1 2 ...
##   ..- attr(*, "names")= chr [1:323] "1" "2" "4" "6" ...
##  $ centers     : num [1:2, 1:8] 1.31 1.21 2.56 2.55 5283.42 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:8] "Channel" "Region" "Fresh" "Milk" ...
```

```
##  $ totss       : num 3.55e+10
##  $ withinss    : num [1:2] 1.39e+10 6.57e+09
##  $ tot.withinss: num 2.04e+10
##  $ betweenss   : num 1.51e+10
##  $ size        : int [1:2] 228 95
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

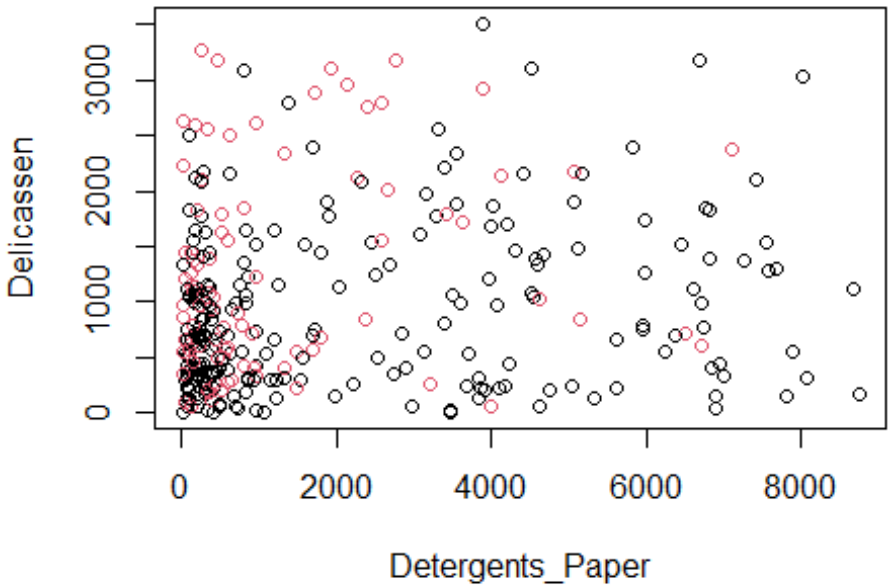
```
#  Fresh y Milk
plot(customers_data2[c(3,4)], col=k2$cluster)
```




```
# Grocery y Frozen
plot(customers_data2[c(5,6)], col=k2$cluster)
```

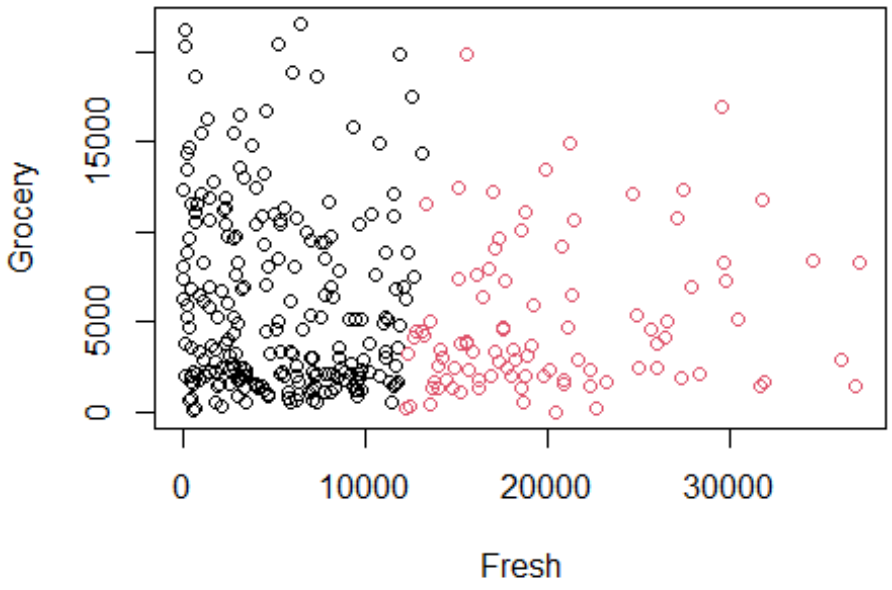



```r
# Detergents_Paper y Delicassen
plot(customers_data2[c(7,8)], col=k2$cluster)
```

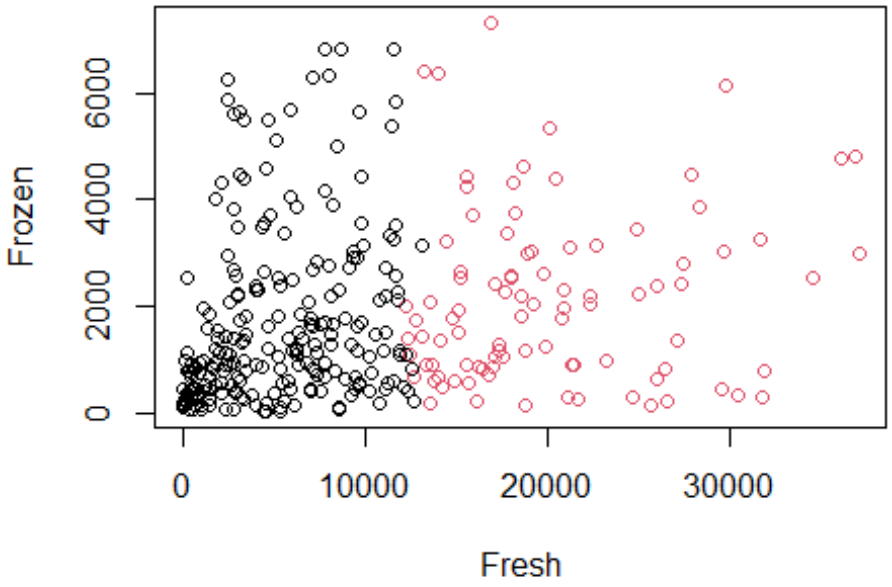


Como se observa en la graficas presentadas anteriormente sólo se puede ver una distinción entre las variables Fresh y Milk, por eso se validara Fesh vs las demás

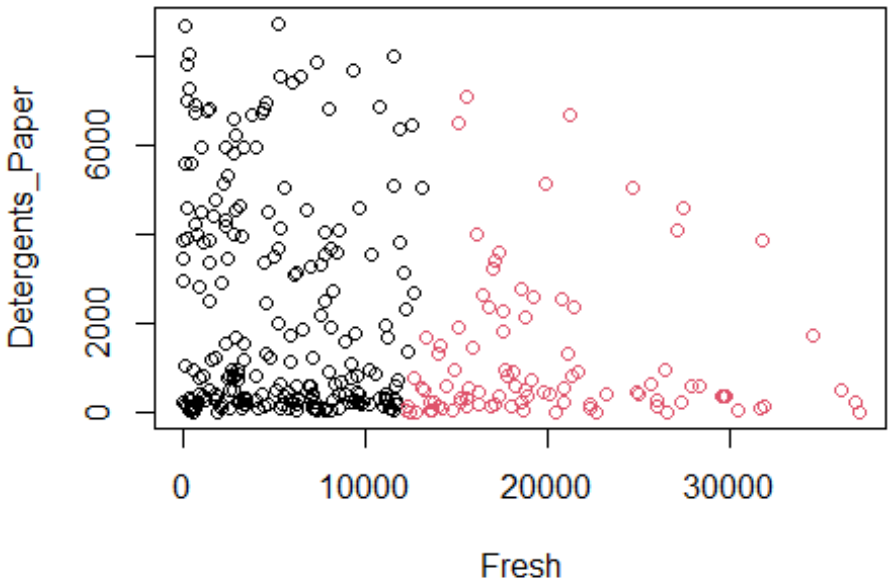
```
#  Fresh y Grocery
plot(customers_data2[c(3,5)], col=k2$cluster)
```



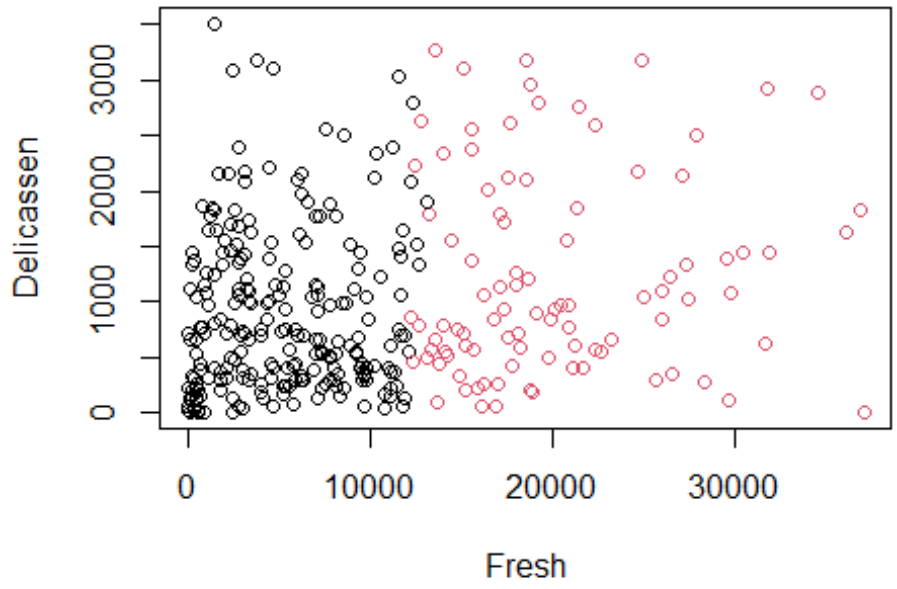

```
# Fresh y Frozen
 plot(customers_data2[c(3,6)], col=k2$cluster)
```




```r
# Fresh y Detergents_Paper
 plot(customers_data2[c(3,7)], col=k2$cluster)
```

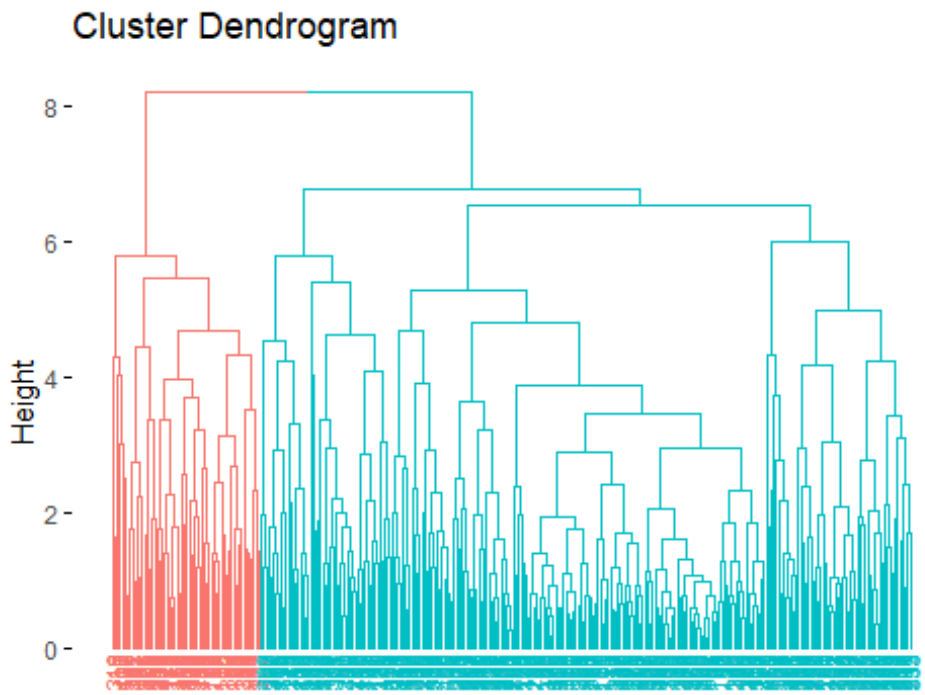


```r
# Fresh y Delicassen
plot(customers_data2[c(3,8)], col=k2$cluster)
```



Como primer método se usará *Hierarchical clustering* el cual agrupa los datos basándose en la distancia entre cada uno de los datos. Con este método no es necesario indicar el número de clusters ya que lo hace de manera jerarquica

```r
#Definición del Hierarchical clustering con método de aglomeración comple
to. Conexión completa: La distancia se mide entre los dos puntos más leja
nos de cada clúster
cluster_hc <- hclust(d = dist(x = customers_data_scale, method = "euclide
an"),
                                method = "complete")
cluster_hc

##
## Call:
## hclust(d = dist(x = customers_data_scale, method = "euclidean"),     m
ethod = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 323

#Se grafica el cluster definido
fviz_dend(x = cluster_hc, k = 2, cex = 0.6)
```



Usando el Método K-medoids clustering - Partitioning Around Medoids (PAM)

Este método es parecido al K-Means la diferencia es que, en K-medoids, cada cluster está representado por una observación presente en el cluster (medoid), mientras que en K-means cada cluster está representado por su centroide, que se corresponde con el promedio de todas las observaciones del cluster pero con ninguna en particular.

medoid es: elemento dentro de un cluster cuya distancia (diferencia) promedio entre él y todos los demás elementos del mismo cluster es lo menor posible

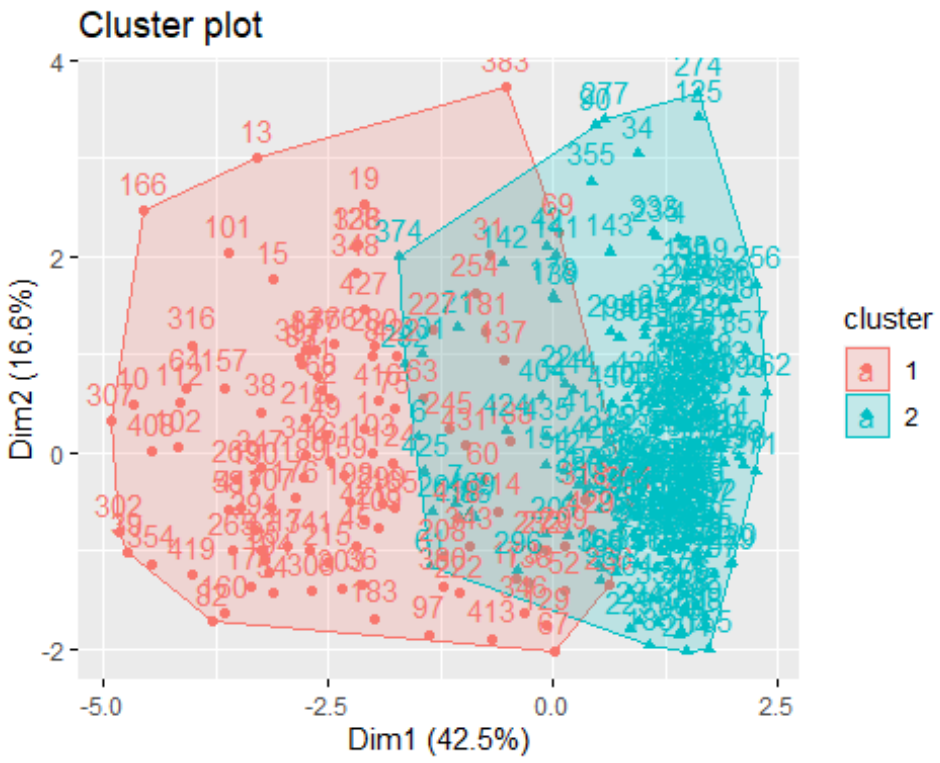
El hecho de utilizar medoids en lugar de centroides hace de K-medoids un método más robusto que K-means, viéndose menos afectado por *outliers* o ruido.

```
#Se genera el cluster PAM con 2
cluster_pam <- pam(x = customers_data2, k = 2, metric = "manhattan")
cluster_pam

## Medoids:
##      ID Channel Region Fresh Milk Grocery Frozen Detergents_Paper Deli
cassen
## 198 146       2      1  2427 7097   10391   1127             4314
1468
## 247 182       1      1  8885 2428    1777   1777              430
610
## Clustering vector:
##   1   2   4   6   7   8   9  10  11  12  13  14  15  16  17  19  20  2
1  22  26
```

```
##   1   1   2   2   2   1   2   1   1   2   1   1   1   2   1   1   2
2   2   2
##  27  28  31  32  33  34  35  36  38  39  42  43  45  49  51  52  54  5
5  56  58
##   2   2   1   2   2   2   2   1   1   1   2   1   1   1   2   1   1
2   2   1
##  59  60  61  63  64  65  67  68  69  70  75  76  79  80  81  82  83  8
4  85  90
##   2   1   2   1   1   2   1   1   1   2   1   2   2   2   2   1   1
2   2   2
##  91  95  96  97  98  99 100 101 102 103 105 106 107 109 111 112 114 11
5 116 117
##   2   1   2   1   2   2   2   1   1   1   2   2   1   1   2   1   2
2   2   2
## 118 119 120 121 122 123 124 125 128 129 131 132 133 134 135 136 137 13
8 139 140
##   2   2   2   2   2   2   1   2   1   1   2   2   2   2   2   2   1
1   2   2
## 141 142 143 145 147 148 149 150 151 152 153 154 155 157 158 159 160 16
1 162 163
##   2   2   2   2   2   2   2   2   2   2   2   2   2   1   2   1   1
1   2   2
## 165 166 168 169 170 171 173 175 176 178 179 180 181 183 185 186 187 18
8 189 190
##   1   1   2   2   2   1   1   2   1   2   2   2   1   1   2   2   2
1   1   1
## 191 192 193 194 195 198 199 200 204 205 207 208 209 211 213 214 215 21
6 218 220
##   2   2   2   1   2   1   2   2   2   2   2   1   1   2   2   1   1
1   2   2
## 221 222 224 225 226 227 228 229 230 232 233 234 235 236 237 238 239 24
2 243 244
##   2   1   2   2   2   1   2   2   2   1   2   2   2   1   2   2   2
2   2   2
## 245 247 248 249 250 251 254 256 257 258 261 262 263 264 265 269 270 27
1 272 273
##   1   2   2   2   2   2   1   2   2   2   2   2   2   1   1   1   2
2   2   1
## 274 275 276 277 279 280 281 282 287 289 291 292 293 294 295 296 297 29
8 299 300
##   2   2   2   2   2   1   2   2   2   2   2   2   2   1   2   2   2
2   1   2
## 301 302 303 306 307 308 309 312 314 315 316 317 318 319 321 322 323 32
4 325 327
##   2   1   1   1   1   2   2   2   2   2   1   2   1   2   2   2   2
2   2   2
## 328 330 331 333 336 337 341 342 343 345 346 347 348 349 351 353 354 35
5 356 357
##   2   2   2   2   1   2   1   1   1   2   1   1   1   2   2   2   1
2   2   2
```

```
## 360 361 362 363 364 365 366 367 368 369 370 374 375 376 377 379 380 38
1 383 384
##   2   2   2   2   2   2   1   2   2   2   2   2   2   2   1   2   1
2   1   2
## 386 387 388 389 390 391 392 393 395 396 397 398 399 400 401 403 404 40
5 406 408
##   2   2   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2
2   2   1
## 409 411 413 415 416 417 418 419 420 421 422 423 424 425 427 429 430 43
1 433 434
##   2   2   1   2   1   1   1   1   2   1   1   2   2   2   1   1   2
1   2   2
## 435 439 440
##   2   2   2
## Objective function:
##    build     swap
## 12972.24 12593.89
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

```
#Se grafica el cluster definido
fviz_cluster(object = cluster_pam, data = customers_data_scale)
```




```
#Descripción de los dos clusters
k2
```

```
## K-means clustering with 2 clusters of sizes 228, 95
##
## Cluster means:
##    Channel   Region     Fresh     Milk  Grocery   Frozen Detergents_Pa
per
## 1 1.307018 2.561404  5283.417 4335.754 6031.583 1682.211         2198.
873
## 2 1.210526 2.547368 20168.853 3512.474 4959.232 2111.758         1189.
158
##   Delicassen
## 1   893.0965
## 2  1218.3684
##
## Clustering vector:
##   1   2   4   6   7   8   9  10  11  12  13  14  15  16  17  19  20  2
1  22  26
##   1   1   2   1   1   1   1   1   1   2   2   2   2   1   1   2   1
2   1   2
##  27  28  31  32  33  34  35  36  38  39  42  43  45  49  51  52  54  5
5  56  58
##   1   2   2   1   2   2   1   1   2   1   2   1   1   1   1   1   1
2   1   1
##  59  60  61  63  64  65  67  68  69  70  75  76  79  80  81  82  83  8
4  85  90
##   2   1   1   1   1   1   1   2   1   1   1   2   1   1   1   1   1
2   1   2
##  91  95  96  97  98  99 100 101 102 103 105 106 107 109 111 112 114 11
5 116 117
##   1   1   1   1   1   1   1   1   1   1   2   2   1   1   1   1   2
2   1   1
## 118 119 120 121 122 123 124 125 128 129 131 132 133 134 135 136 137 13
8 139 140
##   1   2   1   2   1   2   1   2   2   1   1   1   2   1   1   1   1
1   2   1
## 141 142 143 145 147 148 149 150 151 152 153 154 155 157 158 159 160 16
1 162 163
##   2   2   2   2   1   1   1   2   2   1   2   1   1   1   2   1   1
1   2   2
## 165 166 168 169 170 171 173 175 176 178 179 180 181 183 185 186 187 18
8 189 190
##   1   2   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1
1   1   1
## 191 192 193 194 195 198 199 200 204 205 207 208 209 211 213 214 215 21
6 218 220
##   2   2   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1
1   2   1
## 221 222 224 225 226 227 228 229 230 232 233 234 235 236 237 238 239 24
2 243 244
##   2   1   1   1   2   2   1   1   1   1   2   1   2   1   1   2   1
2   2   1
```

```
## 245 247 248 249 250 251 254 256 257 258 261 262 263 264 265 269 270 27
1 272 273
##   1   1   2   2   1   1   2   2   1   1   1   1   2   1   1   1   2
1   1   1
## 274 275 276 277 279 280 281 282 287 289 291 292 293 294 295 296 297 29
8 299 300
##   2   1   1   2   1   2   1   1   1   2   1   1   1   1   2   1   2
1   1   1
## 301 302 303 306 307 308 309 312 314 315 316 317 318 319 321 322 323 32
4 325 327
##   2   1   1   1   1   2   1   2   1   1   1   1   1   1   1   1   2
2   2   1
## 328 330 331 333 336 337 341 342 343 345 346 347 348 349 351 353 354 35
5 356 357
##   1   1   1   2   2   2   1   1   1   1   1   1   2   1   1   1   1
2   1   2
## 360 361 362 363 364 365 366 367 368 369 370 374 375 376 377 379 380 38
1 383 384
##   1   2   1   1   1   1   1   1   1   2   1   2   1   1   1   1   1
2   2   1
## 386 387 388 389 390 391 392 393 395 396 397 398 399 400 401 403 404 40
5 406 408
##   1   1   2   1   1   1   1   1   2   1   1   1   1   1   1   2   2
2   1   1
## 409 411 413 415 416 417 418 419 420 421 422 423 424 425 427 429 430 43
1 433 434
##   1   1   1   1   1   1   1   1   1   1   2   2   2   2   1   1   1
1   2   1
## 435 439 440
##   2   1   1
##
## Within cluster sum of squares by cluster:
## [1] 13858574305  6574079992
##  (between_SS / total_SS =  42.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.w
ithinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

6. Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

Respecto a las graficas presentadas anteriormente y los datos arrojados por cada cluster podría definir dos asociaciones:

1. Clientes con preferencias por productos frescos.
2. Clientes sin preferencias definidas.

Ahora se mira detalladamente la información de los dos clusters

Se puede concluir que la variable Fresh es la que más diferencia presenta entre el cluster 1 y 2 ya que para el 1 su media es 5283.417 y para el 2 es 20168.853 y una diferencia más pequeña para la variable Delicassen 893.0965 y 1218.3684 respectivamente.

Adicionalmente tenemos una medida de suma de cuadrados del 42% la cual no es muy buena ya que son pocas las variables que hacen una clara discriminación entre los clusters.

```
#Generación del archivo csv con la limpieza de datos

write.csv(customers_data2, file="Wholesale customers data Final.csv")
```

7. Referencias:

- Calvo M., Subirats L., Pérez D. (2019). Introducción a la limpieza y análisis de los datos.
- Megan Squire (2015). Clean Data. Packt Publishing Ltd.

8. Participación de cada Integrante del Grupo:

Contribuciones	Firma
Investigación Previa	LLTB, JCSV
Redacción de las Respuestas	LLTB, JCSV
Desarrollo Código	LLTB, JCSV